Swing Programming Part 1

Suman Kumar

Computer Science Dept.

Troy University

Introduction to Graphics

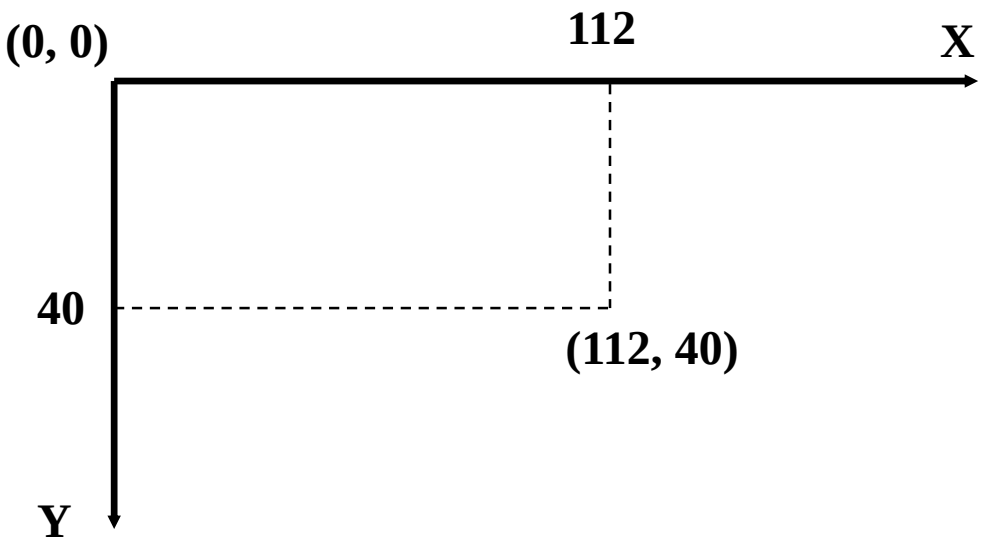
- The last few sections of each chapter of the textbook focus on graphics and graphical user interfaces
- A picture or drawing must be digitized for storage on a computer
- A picture is made up of *pixels* (picture elements), and each pixel is stored separately
- The number of pixels used to represent a picture is called the *picture resolution*
- The number of pixels that can be displayed by a monitor is called the *monitor resolution*

Representing Images

- A digitized picture with a small portion magnified:

Coordinate Systems

- Each pixel can be identified using a two-dimensional coordinate system
- When referring to a pixel in a Java program, we use a coordinate system with the origin in the top-left corner



Representing Color

- A black and white picture could be stored using one bit per pixel (0 = white and 1 = black)
- A colored picture requires more information; there are several techniques for representing colors
- Every color can be represented as a mixture of the three additive primary colors Red, Green, and Blue
- Each color is represented by three numbers between 0 and 255 that collectively are called an *RGB value*

The Color Class

- A color in a Java program is represented as an object created from the `Color` class

- The `Color` class also contains several predefined colors, including the following:

Object	RGB Value
`Color.black`	`0, 0, 0`
`Color.blue`	`0, 0, 255`
`Color.cyan`	`0, 255, 255`
`Color.orange`	`255, 200, 0`
`Color.white`	`255, 255, 255`
`Color.yellow`	`255, 255, 0`

GUI Components

- A *GUI component* is an object that represents a screen element such as a button or a text field
- GUI-related classes are defined primarily in the `java.awt` and the `javax.swing` packages
- The *Abstract Windowing Toolkit* (AWT) was the original Java GUI package
- The *Swing* package provides additional and more versatile components
- Both packages are needed to create a Java GUI-

Copyright © 2014 Pearson Education, Inc.

GUI Containers

- A *GUI container* is a component that is used to hold and organize other components
- A *frame* is a container displayed as a separate window with a title bar
- It can be repositioned and resized on the screen as needed
- A *panel* is a container that cannot be displayed on its own but is used to organize other components
- A panel must be added to another container (like a frame or another panel) to be displayed

Copyright © 2014 Pearson Education, Inc.

GUI Containers

- A GUI container can be classified as either heavyweight or lightweight
- A *heavyweight container* is one that is managed by the underlying operating system
- A *lightweight container* is managed by the Java program itself
- Occasionally this distinction is important
- A frame is a heavyweight container and a panel is a lightweight container

Copyright © 2014 Pearson Education, Inc.

Labels

- A *label* is a GUI component that displays a line of text and/or an image
- Labels are usually used to display information or identify other components in the interface
- Let's look at a program that organizes two labels in a panel and displays that panel in a frame
- This program is not interactive, but the frame can be repositioned and resized
- See `Authority.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Authority.java       Author: Lewis/Loftus
//
//  Demonstrates the use of frames, panels, and labels.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class Authority
{
   //-----------------------------------------------------------------
   //  Displays some words of wisdom.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Authority");

      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JPanel primary = new JPanel();
      primary.setBackground(Color.yellow);
      primary.setPreferredSize(new Dimension(250, 75));
```

continued

Copyright © 2014 Pearson Education, Inc.

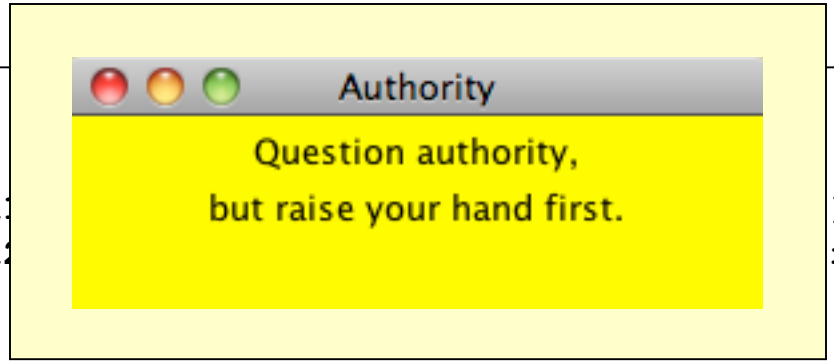
continued

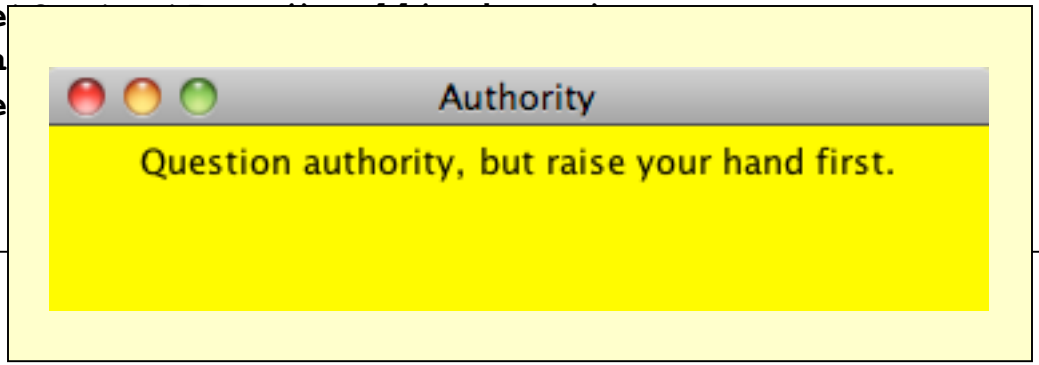
```
        JLabel label1 = new JLabel("Question authority,");
        JLabel label2 = new JLabel("but raise your hand first.");

        primary.add(label1);
        primary.add(label2);

        frame.getContentPane().add(primary);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

continued

```
        JLabel label1
        JLabel label2
                                                    );
                                                    first.");

        primary.add
        primary.add(label2);

        frame.ge
        frame.pa
        frame.se
    }
}
```





Copyright © 2014 Pearson Education, Inc.

Nested Panels

- Containers that contain other components make up the *containment hierarchy* of an interface
- This hierarchy can be as intricate as needed to create the visual effect desired
- The following example nests two panels inside a third panel – note the effect this has as the frame is resized
- See `NestedPanels.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  NestedPanels.java       Author: Lewis/Loftus
//
//  Demonstrates a basic componenet hierarchy.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class NestedPanels
{
   //-----------------------------------------------------------------
   //  Presents two colored panels nested within a third.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Nested Panels");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      // Set up first subpanel
      JPanel subPanel1 = new JPanel();
      subPanel1.setPreferredSize(new Dimension(150, 100));
      subPanel1.setBackground(Color.green);
      JLabel label1 = new JLabel("One");
      subPanel1.add(label1);
```

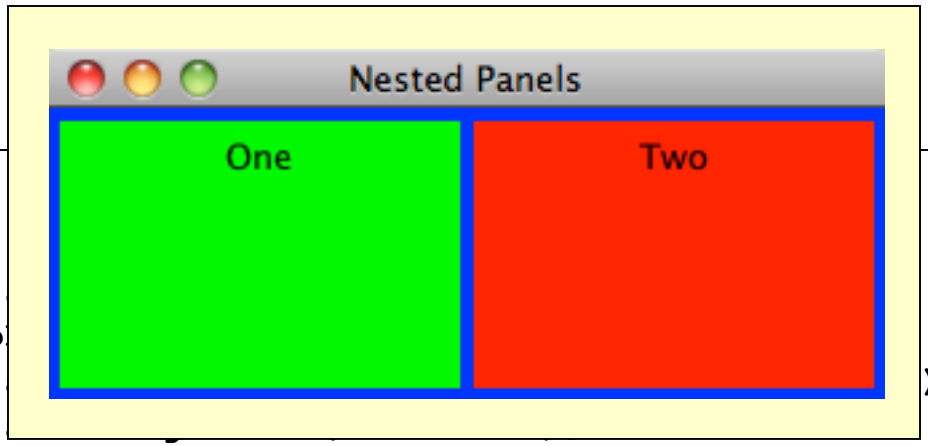
continued

Copyright © 2014 Pearson Education, Inc.

continued

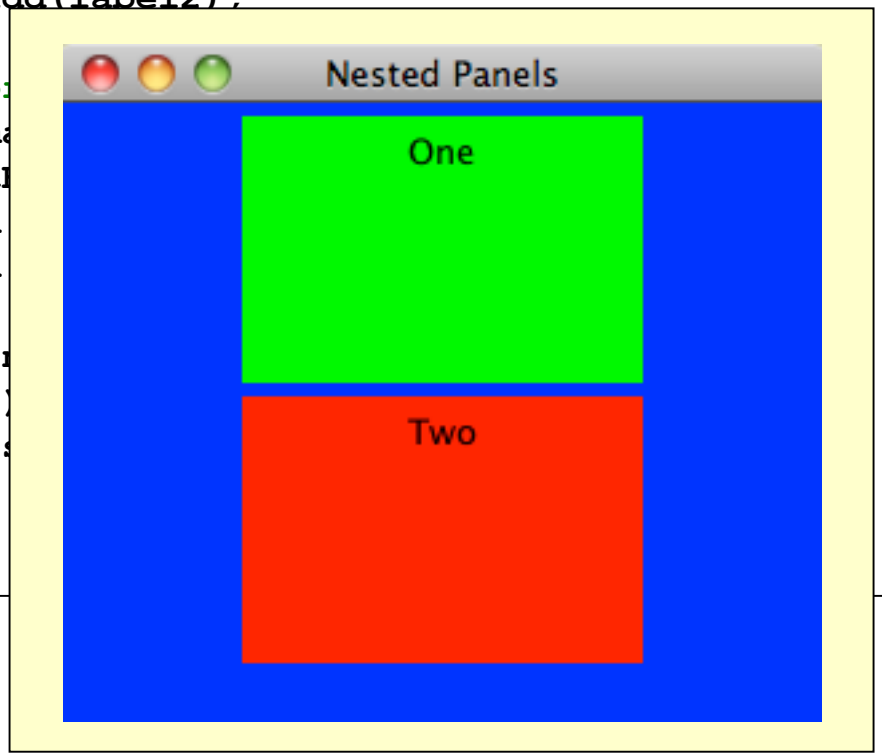
```
        // Set up second subpanel
        JPanel subPanel2 = new JPanel();
        subPanel2.setPreferredSize(new Dimension(150, 100));
        subPanel2.setBackground(Color.red);
        JLabel label2 = new JLabel("Two");
        subPanel2.add(label2);

        // Set up primary panel
        JPanel primary = new JPanel();
        primary.setBackground(Color.blue);
        primary.add(subPanel1);
        primary.add(subPanel2);

        frame.getContentPane().add(primary);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.



continued

```
        // Set up
        JPanel sub
        subPanel2.                                          );
        subPanel2.
        JLabel label2 = new JLabel("Two");
        subPanel2.add(label2);

        // Set up pr
        JPanel prima
        primary.setL
        primary.add
        primary.add

        frame.getCo
        frame.pack()
        frame.setVis
    }
}
```



Copyright © 2014 Pearson Education, Inc.

Outline

- Creating Objects
- The String Class
- The Random and Math Classes
- Formatting Output
- Enumerated Types
- Wrapper Classes
- Components and Containers
- Images

Copyright © 2014 Pearson Education, Inc.

Images

- Images can be displayed in a Java program in various ways
- As we've seen, a `JLabel` object can be used to display a line of text
- It can also be used to display an image
- That is, a label can be composed of text, an image, or both at the same time

Copyright © 2014 Pearson Education, Inc.

Images

- The `ImageIcon` class is used to represent the image that is stored in a label
- If text is also included, the position of the text relative to the image can be set explicitly
- The alignment of the text and image within the label can be set as well
- See `LabelDemo.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  LabelDemo.java       Author: Lewis/Loftus
//
//  Demonstrates the use of image icons in labels.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class LabelDemo
{
   //-----------------------------------------------------------------
   //  Creates and displays the primary application frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Label Demo");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      ImageIcon icon = new ImageIcon("devil.gif");

      JLabel label1, label2, label3;

      label1 = new JLabel("Devil Left", icon, SwingConstants.CENTER);
```

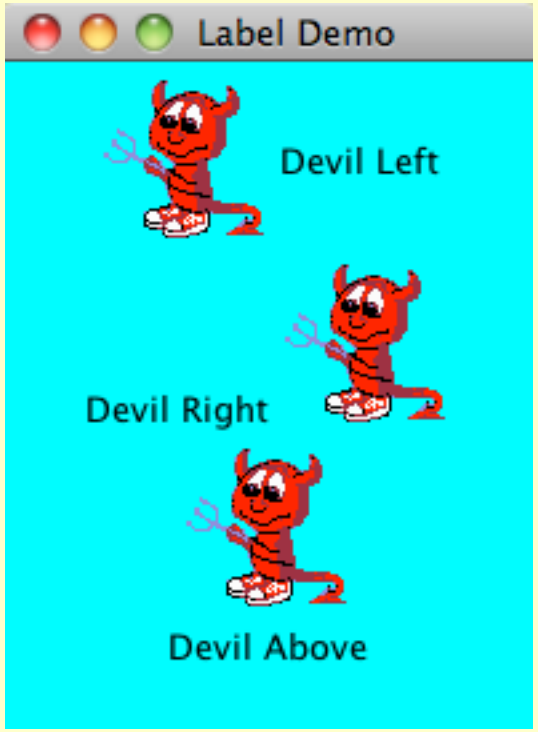
continued

continued

```
        label2 = new JLabel("Devil Right", icon, SwingConstants.CENTER);
        label2.setHorizontalTextPosition(SwingConstants.LEFT);
        label2.setVerticalTextPosition(SwingConstants.BOTTOM);

        label3 = new JLabel("Devil Above", icon, SwingConstants.CENTER);
        label3.setHorizontalTextPosition(SwingConstants.CENTER);
        label3.setVerticalTextPosition(SwingConstants.BOTTOM);

        JPanel panel = new JPanel();
        panel.setBackground(Color.cyan);
        panel.setPreferredSize(new Dimension(200, 250));
        panel.add(label1);
        panel.add(label2);
        panel.add(label3);

        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
```

continued

```
        label2 = new JL                                    ingConstants.CENTER);
        label2.setHori                                     ants.LEFT);
        label2.setVerti                                    ts.BOTTOM);

        label3 = new JL                                    ingConstants.CENTER);
        label3.setHori                                     ants.CENTER);
        label3.setVerti                                    ts.BOTTOM);

        JPanel panel =
        panel.setBackg
        panel.setPrefe                                     250));
        panel.add(label
        panel.add(label
        panel.add(label

        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
```



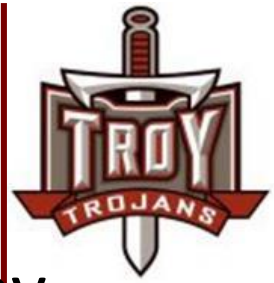
Copyright © 2014 Pearson Education, Inc.

Graphical Objects

- Some objects contain information that determines how the object should be represented visually
- Most GUI components are graphical objects
- We can have some effect on how components get drawn
- We did this in Chapter 2 when we defined the `paint` method of an applet
- Let's look at some other examples of graphical objects

Copyright © 2014 Pearson Education, Inc.

Smiling Face Example



- The `SmilingFace` program draws a face by defining the `paintComponent` method of a panel
- See `SmilingFace.java`
- See `SmilingFacePanel.java`
- The `main` method of the `SmilingFace` class instantiates a `SmilingFacePanel` and displays it
- The `SmilingFacePanel` class is derived from the `JPanel` class using inheritance

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  SmilingFace.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a separate panel class.
//********************************************************************

import javax.swing.JFrame;

public class SmilingFace
{
   //-----------------------------------------------------------------
   //  Creates the main frame of the program.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Smiling Face");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      SmilingFacePanel panel = new SmilingFacePanel();

      frame.getContentPane().add(panel);

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//************
//  SmilingFac
//
//  Demonstrat
//************

import javax.s

public class S
{
   //--------
   //  Creates
   //--------
   public stat
   {
      JFrame frame = new JFrame("Smiling Face");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      SmilingFacePanel panel = new SmilingFacePanel();

      frame.getContentPane().add(panel);

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  SmilingFacePanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of a separate panel class.
//********************************************************************

import javax.swing.JPanel;
import java.awt.*;

public class SmilingFacePanel extends JPanel
{
   private final int BASEX = 120, BASEY = 60; // base point for head

   //-----------------------------------------------------------------
   //  Constructor: Sets up the main characteristics of this panel.
   //-----------------------------------------------------------------
   public SmilingFacePanel()
   {
      setBackground(Color.blue);
      setPreferredSize(new Dimension(320, 200));
      setFont(new Font("Arial", Font.BOLD, 16));
   }
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Draws a face.
//-----------------------------------------------------------------
public void paintComponent(Graphics page)
{
   super.paintComponent(page);

   page.setColor(Color.yellow);
   page.fillOval(BASEX, BASEY, 80, 80);  // head
   page.fillOval(BASEX-5, BASEY+20, 90, 40);  // ears

   page.setColor(Color.black);
   page.drawOval(BASEX+20, BASEY+30, 15, 7);  // eyes
   page.drawOval(BASEX+45, BASEY+30, 15, 7);

   page.fillOval(BASEX+25, BASEY+31, 5, 5);   // pupils
   page.fillOval(BASEX+50, BASEY+31, 5, 5);

   page.drawArc(BASEX+20, BASEY+25, 15, 7, 0, 180);  // eyebrows
   page.drawArc(BASEX+45, BASEY+25, 15, 7, 0, 180);

   page.drawArc(BASEX+35, BASEY+40, 15, 10, 180, 180);  // nose
   page.drawArc(BASEX+20, BASEY+50, 40, 15, 180, 180);  // mouth
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
        page.setColor(Color.white);
        page.drawString("Always remember that you are unique!",
                        BASEX-105, BASEY-15);
        page.drawString("Just like everyone else.", BASEX-45, BASEY+105);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

Smiling Face Example

- Every Swing component has a `paintComponent` method
- The `paintComponent` method accepts a `Graphics` object that represents the graphics context for the panel
- We define the `paintComponent` method to draw the face with appropriate calls to the `Graphics` methods
- Note the difference between drawing on a panel and adding other GUI components to a panel

Copyright © 2014 Pearson Education, Inc.

Splat Example

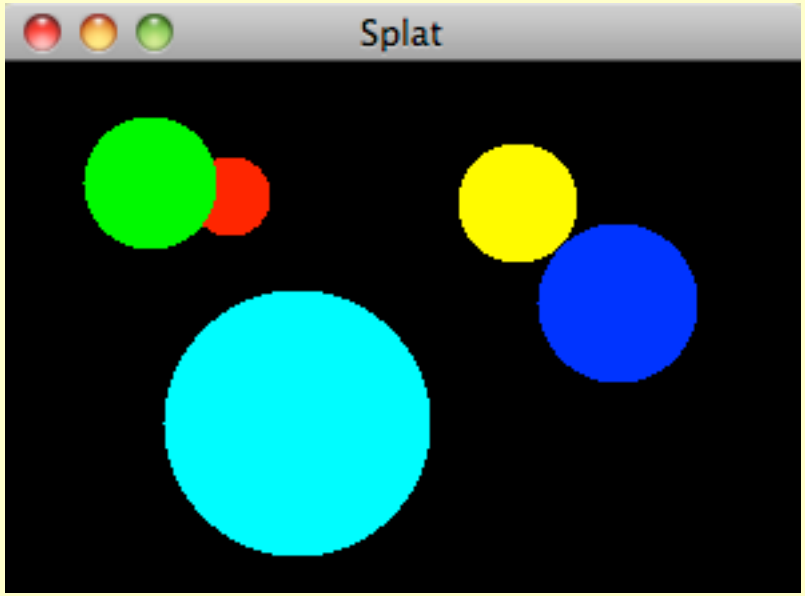
- The `Splat` example is structured a bit differently
- It draws a set of colored circles on a panel, but each circle is represented as a separate object that maintains its own graphical information
- The `paintComponent` method of the panel "asks" each circle to draw itself
- See `Splat.java`
- See `SplatPanel.java`
- See `Circle.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Splat.java       Author: Lewis/Loftus
//
//  Demonstrates the use of graphical objects.
//********************************************************************

import javax.swing.*;
import java.awt.*;

public class Splat
{
   //-----------------------------------------------------------------
   //  Presents a collection of circles.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Splat");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new SplatPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//*************
//  Splat.java
//
//  Demonstrate
//*************

import javax.sw
import java.awt

public class Sp
{
   //---------
   //  Presents
   //---------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Splat");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new SplatPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```



Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  SplatPanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of graphical objects.
//********************************************************************

import javax.swing.*;
import java.awt.*;

public class SplatPanel extends JPanel
{
   private Circle circle1, circle2, circle3, circle4, circle5;

   //-----------------------------------------------------------------
   //  Constructor: Creates five Circle objects.
   //-----------------------------------------------------------------
   public SplatPanel()
   {
      circle1 = new Circle(30, Color.red, 70, 35);
      circle2 = new Circle(50, Color.green, 30, 20);
      circle3 = new Circle(100, Color.cyan, 60, 85);
      circle4 = new Circle(45, Color.yellow, 170, 30);
      circle5 = new Circle(60, Color.blue, 200, 60);

      setPreferredSize(new Dimension(300, 200));
      setBackground(Color.black);
   }
```

continue

continue

```
    //-----------------------------------------------------------------
    //  Draws this panel by requesting that each circle draw itself.
    //-----------------------------------------------------------------
    public void paintComponent(Graphics page)
    {
        super.paintComponent(page);

        circle1.draw(page);
        circle2.draw(page);
        circle3.draw(page);
        circle4.draw(page);
        circle5.draw(page);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Circle.java       Author: Lewis/Loftus
//
//  Represents a circle with a particular position, size, and color.
//********************************************************************

import java.awt.*;

public class Circle
{
   private int diameter, x, y;
   private Color color;

   //-----------------------------------------------------------------
   //  Constructor: Sets up this circle with the specified values.
   //-----------------------------------------------------------------
   public Circle(int size, Color shade, int upperX, int upperY)
   {
      diameter = size;
      color = shade;
      x = upperX;
      y = upperY;
   }
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Draws this circle in the specified graphics context.
//-----------------------------------------------------------------
public void draw(Graphics page)
{
   page.setColor(color);
   page.fillOval(x, y, diameter, diameter);
}

//-----------------------------------------------------------------
//  Diameter mutator.
//-----------------------------------------------------------------
public void setDiameter(int size)
{
   diameter = size;
}

//-----------------------------------------------------------------
//  Color mutator.
//-----------------------------------------------------------------
public void setColor(Color shade)
{
   color = shade;
}
```

continue

Education, Inc.

continue

```
//-----------------------------------------------------------------
//  X mutator.
//-----------------------------------------------------------------
public void setX(int upperX)
{
   x = upperX;
}

//-----------------------------------------------------------------
//  Y mutator.
//-----------------------------------------------------------------
public void setY(int upperY)
{
   y = upperY;
}

//-----------------------------------------------------------------
//  Diameter accessor.
//-----------------------------------------------------------------
public int getDiameter()
{
   return diameter;
}
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
    //-----------------------------------------------------------------
    //  Color accessor.
    //-----------------------------------------------------------------
    public Color getColor()
    {
       return color;
    }

    //-----------------------------------------------------------------
    //  X accessor.
    //-----------------------------------------------------------------
    public int getX()
    {
       return x;
    }

    //-----------------------------------------------------------------
    //  Y accessor.
    //-----------------------------------------------------------------
    public int getY()
    {
       return y;
    }
}
```

Copyright © 2014 Pearson Education, Inc.

Outline

- Anatomy of a Class
- Encapsulation
- Anatomy of a Method
- Graphical Objects
- Graphical User Interfaces
- Buttons and Text Fields

Copyright © 2014 Pearson Education, Inc.

Graphical User Interfaces

- A Graphical User Interface (GUI) in Java is created with at least three kinds of objects:
  - components, events, and listeners
- We've previously discussed *components*, which are objects that represent screen elements:
  - labels, buttons, text fields, menus, etc.
- Some components are *containers* that hold and organize other components:
  - frames, panels, applets, dialog boxes

Copyright © 2014 Pearson Education, Inc.

Events

- An *event* is an object that represents some activity to which we may want to respond
- For example, we may want our program to perform some action when the following occurs:
  - the mouse is moved
  - the mouse is dragged
  - a mouse button is clicked
  - a graphical button is pressed
  - a keyboard key is pressed
  - a timer expires

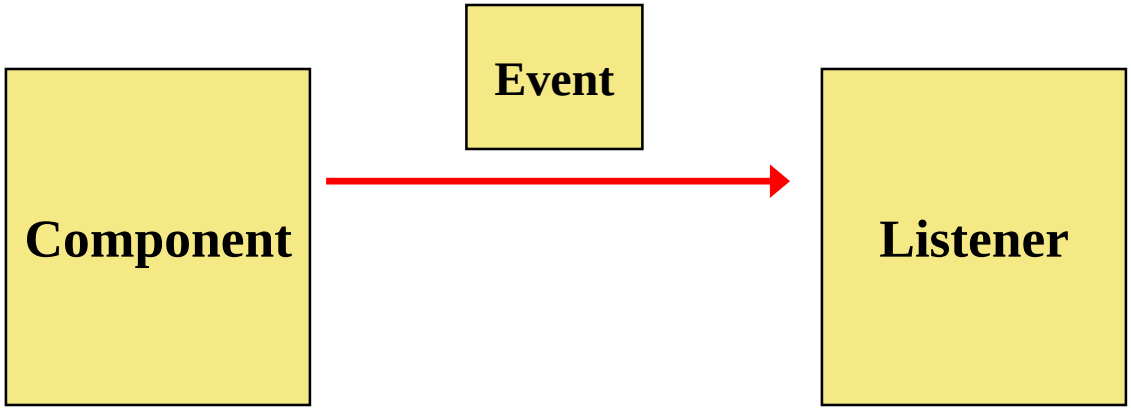
Copyright © 2014 Pearson Education, Inc.

Events and Listeners

- The Java API contains several classes that represent typical events
- Components, such as a graphical button, generate (or fire) an event when it occurs
- We set up a *listener* object to respond to an event when it occurs
- We can design listener objects to take whatever actions are appropriate when an event occurs

Copyright © 2014 Pearson Education, Inc.

Events and Listeners



A component object generates an event

A corresponding listener object is designed to respond to the event

When the event occurs, the component calls the appropriate method of the listener, passing an object that describes the event

Copyright © 2014 Pearson Education, Inc.

GUI Development

- To create a Java program that uses a GUI we must:
  - instantiate and set up the necessary components
  - implement listener classes for any events we care about
  - establish the relationship between listeners and the components that generate the corresponding events
- Let's now explore some new components and see how this all comes together

Copyright © 2014 Pearson Education, Inc.

Outline

- Anatomy of a Class
- Encapsulation
- Anatomy of a Method
- Graphical Objects
- Graphical User Interfaces
- Buttons and Text Fields

Copyright © 2014 Pearson Education, Inc.

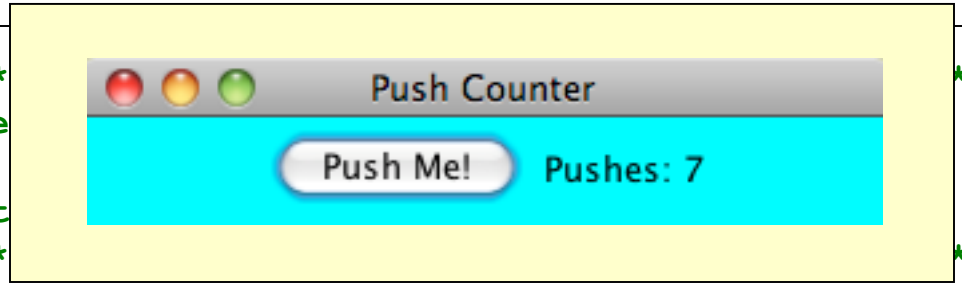
Buttons

- A *push button* is defined by the `JButton` class
- It generates an *action event*
- The `PushCounter` example displays a push button that increments a counter each time it is pushed
- See `PushCounter.java`
- See `PushCounterPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  PushCounter.java       Authors: Lewis/Loftus
//
//  Demonstrates a graphical user interface and an event listener.
//********************************************************************

import javax.swing.JFrame;

public class PushCounter
{
   //-----------------------------------------------------------------
   //  Creates the main program frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Push Counter");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new PushCounterPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.



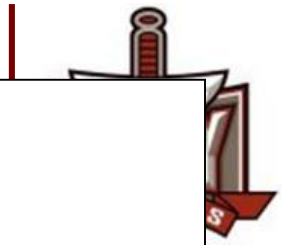

```
//*************                                              **************
//  PushCounte
//
//  Demonstrat                                               listener.
//*************                                              **************

import javax.swing.JFrame;

public class PushCounter
{
   //-----------------------------------------------------------------
   //  Creates the main program frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Push Counter");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      frame.getContentPane().add(new PushCounterPanel());

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  PushCounterPanel.java       Authors: Lewis/Loftus
//
//  Demonstrates a graphical user interface and an event listener.
//********************************************************************

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class PushCounterPanel extends JPanel
{
   private int count;
   private JButton push;
   private JLabel label;

   //-----------------------------------------------------------------
   //  Constructor: Sets up the GUI.
   //-----------------------------------------------------------------
   public PushCounterPanel()
   {
      count = 0;

      push = new JButton("Push Me!");
      push.addActionListener(new ButtonListener());
```

continue

Copyright © 2014 Pearson Education, Inc.



continue

```
        label = new JLabel("Pushes: " + count);

        add(push);
        add(label);

        setPreferredSize(new Dimension(300, 40));
        setBackground(Color.cyan);
    }

    //*****************************************************************
    //  Represents a listener for button push (action) events.
    //*****************************************************************
    private class ButtonListener implements ActionListener
    {
        //--------------------------------------------------------------
        //  Updates the counter and label when the button is pushed.
        //--------------------------------------------------------------
        public void actionPerformed(ActionEvent event)
        {
            count++;
            label.setText("Pushes: " + count);
        }
    }
}
```

Copyright © 2014 Pearson Education, Inc.

Push Counter Example

- The components of the GUI are the button, a label to display the counter, a panel to organize the components, and the main frame
- The `PushCounterPanel` class represents the panel used to display the button and label
- The `PushCounterPanel` class is derived from `JPanel` using inheritance
- The constructor of `PushCounterPanel` sets up the elements of the GUI and initializes the counter to zero

Copyright © 2014 Pearson Education, Inc.

Push Counter Example

- The `ButtonListener` class is the listener for the action event generated by the button
- It is implemented as an *inner class*, which means it is defined within the body of another class
- That facilitates the communication between the listener and the GUI components
- Inner classes should only be used in situations where there is an intimate relationship between the two classes and the inner class is not needed in any other context

Copyright © 2014 Pearson Education, Inc.

Push Counter Example

- Listener classes are written by implementing a *listener interface*
- The `ButtonListener` class implements the `ActionListener` interface
- An interface is a list of methods that the implementing class must define
- The only method in the `ActionListener` interface is the `actionPerformed` method
- The Java API contains interfaces for many types of events
- We discuss interfaces in more detail in Chapter 6

Copyright © 2014 Pearson Education, Inc.

Push Counter Example

- The `PushCounterPanel` constructor:
  - instantiates the `ButtonListener` object
  - establishes the relationship between the button and the listener by the call to `addActionListener`
- When the user presses the button, the button component creates an `ActionEvent` object and calls the `actionPerformed` method of the listener
- The `actionPerformed` method increments the counter and resets the text of the label

Copyright © 2014 Pearson Education, Inc.

Quick Check

Which object in the Push Counter example generated the event?

What did it do then?

Copyright © 2014 Pearson Education, Inc.

Quick Check

Which object in the Push Counter example generated the event?

The button component generated the event.

What did it do then?

It called the `actionPerformed` method of the listener object that had been registered with it.

Copyright © 2014 Pearson Education, Inc.

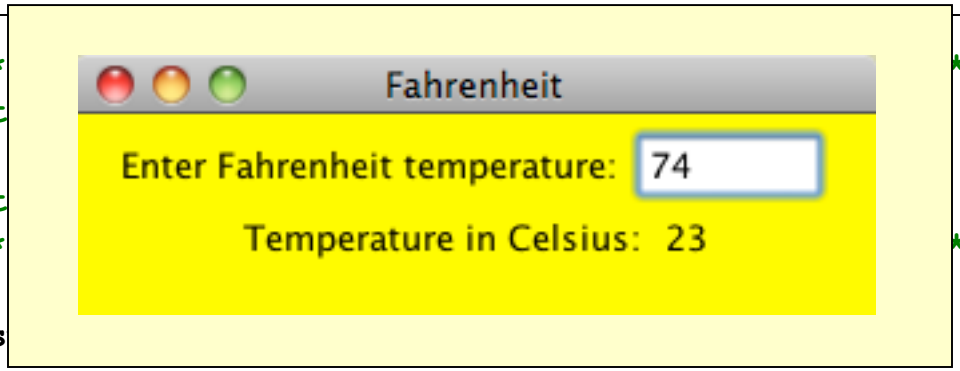
Text Fields

- Let's look at another GUI example that uses another type of component
- A *text field* allows the user to enter one line of input
- If the cursor is in the text field, the text field object generates an action event when the enter key is pressed
- See `Fahrenheit.java`
- See `FahrenheitPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Fahrenheit.java       Author: Lewis/Loftus
//
//  Demonstrates the use of text fields.
//********************************************************************

import javax.swing.JFrame;

public class Fahrenheit
{
   //-----------------------------------------------------------------
   //  Creates and displays the temperature converter GUI.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Fahrenheit");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      FahrenheitPanel panel = new FahrenheitPanel();

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//************                                          **************
//  Fahrenheit
//
//  Demonstrat
//************                                          **************

import javax.s

public class Fahrenheit
{
   //-----------------------------------------------------------------
   //  Creates and displays the temperature converter GUI.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Fahrenheit");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      FahrenheitPanel panel = new FahrenheitPanel();

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  FahrenheitPanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of text fields.
//********************************************************************

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class FahrenheitPanel extends JPanel
{
   private JLabel inputLabel, outputLabel, resultLabel;
   private JTextField fahrenheit;

   //-----------------------------------------------------------------
   //  Constructor: Sets up the main GUI components.
   //-----------------------------------------------------------------
   public FahrenheitPanel()
   {
      inputLabel = new JLabel("Enter Fahrenheit temperature:");
      outputLabel = new JLabel("Temperature in Celsius: ");
      resultLabel = new JLabel("---");

      fahrenheit = new JTextField(5);
      fahrenheit.addActionListener(new TempListener());
```

continue

Education, Inc.

continue

```
        add(inputLabel);
        add(fahrenheit);
        add(outputLabel);
        add(resultLabel);

        setPreferredSize(new Dimension(300, 75));
        setBackground(Color.yellow);
    }

    //*****************************************************************
    //  Represents an action listener for the temperature input field.
    //*****************************************************************
    private class TempListener implements ActionListener
    {
        //-----------------------------------------------------------
        //  Performs the conversion when the enter key is pressed in
        //  the text field.
        //-----------------------------------------------------------
        public void actionPerformed(ActionEvent event)
        {
            int fahrenheitTemp, celsiusTemp;

            String text = fahrenheit.getText();
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
            fahrenheitTemp = Integer.parseInt(text);
            celsiusTemp = (fahrenheitTemp-32) * 5/9;

            resultLabel.setText(Integer.toString(celsiusTemp));
        }
    }
}
```

Copyright © 2014 Pearson Education, Inc.

Check Boxes

- A *check box* is a button that can be toggled on or off
- It is represented by the `JCheckBox` class
- Unlike a push button, which generates an action event, a check box generates an *item event* whenever it changes state
- The `ItemListener` interface is used to define item event listeners
- A check box calls the `itemStateChanged` method of the listener when it is toggled

Copyright © 2014 Pearson Education, Inc.

Check Boxes

- Let's examine a program that uses check boxes to determine the style of a label's text string
- It uses the `Font` class, which embodies a character font's:
  - family name (such as Times or Courier)
  - style (bold, italic, or both)
  - font size
- See `StyleOptions.java`
- See `StyleOptionsPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  StyleOptions.java       Author: Lewis/Loftus
//
//  Demonstrates the use of check boxes.
//********************************************************************

import javax.swing.JFrame;

public class StyleOptions
{
   //-----------------------------------------------------------------
   //  Creates and presents the program frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Style Options");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      StyleOptionsPanel panel = new StyleOptionsPanel();
      frame.getContentPane().add(panel);

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson
Education, Inc.

```
//*****************                                          *************
//  StyleOptions.
//
//  Demonstrates
//*****************                                          *************

import javax.swin

public class StyleOptions
{
   //-------------                                          -------------
   //  Creates an
   //-------------                                          -------------
   public static
   {
      JFrame fram
      frame.setDe                                                  SE);

      StyleOption
      frame.getContentPane().add(panel);

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  StyleOptionsPanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of check boxes.
//********************************************************************

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class StyleOptionsPanel extends JPanel
{
   private JLabel saying;
   private JCheckBox bold, italic;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Sets up a panel with a label and some check boxes that
//  control the style of the label's font.
//-----------------------------------------------------------------
public StyleOptionsPanel()
{
   saying = new JLabel("Say it with style!");
   saying.setFont(new Font("Helvetica", Font.PLAIN, 36));

   bold = new JCheckBox("Bold");
   bold.setBackground(Color.cyan);
   italic = new JCheckBox("Italic");
   italic.setBackground(Color.cyan);

   StyleListener listener = new StyleListener();
   bold.addItemListener(listener);
   italic.addItemListener(listener);

   add(saying);
   add(bold);
   add(italic);

   setBackground(Color.cyan);
   setPreferredSize(new Dimension(300, 100));
}
```

continue

continue

```
   //*****************************************************************
   //  Represents the listener for both check boxes.
   //*****************************************************************
   private class StyleListener implements ItemListener
   {
      //--------------------------------------------------------------
      //  Updates the style of the label font style.
      //--------------------------------------------------------------
      public void itemStateChanged(ItemEvent event)
      {
         int style = Font.PLAIN;

         if (bold.isSelected())
            style = Font.BOLD;

         if (italic.isSelected())
            style += Font.ITALIC;

         saying.setFont(new Font("Helvetica", style, 36));
      }
   }
}
```

Copyright © 2014 Pearson Education, Inc.

Radio Buttons

- A group of *radio buttons* represents a set of mutually exclusive options – only one can be selected at any given time
- When a radio button from a group is selected, the button that is currently "on" in the group is automatically toggled off
- To define the group of radio buttons that will work together, each radio button is added to a `ButtonGroup` object
- A radio button generates an action event

Copyright © 2014 Pearson Education, Inc.

Radio Buttons

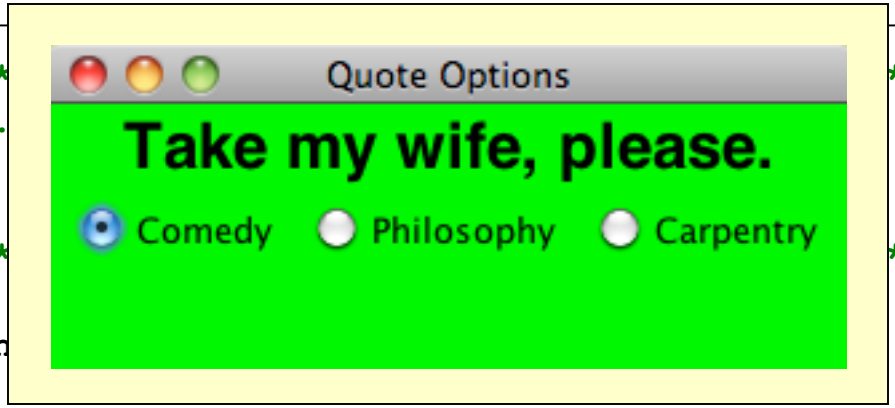
- Let's look at a program that uses radio buttons to determine which line of text to display

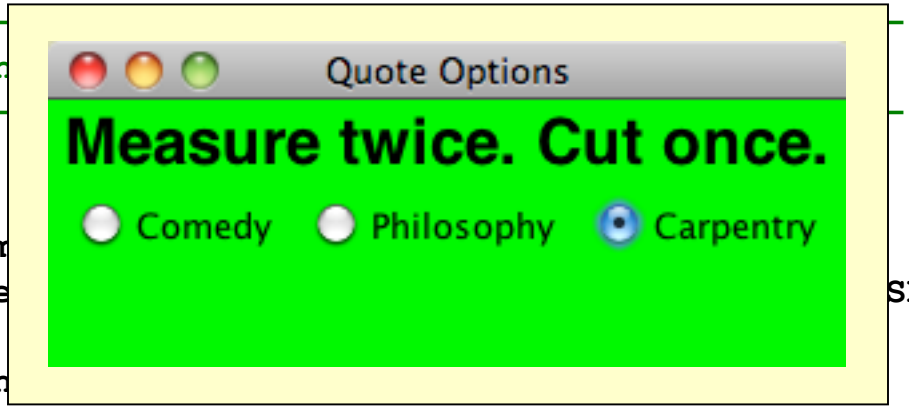
- See `QuoteOptions.java`
- See `QuoteOptionsPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  QuoteOptions.java       Author: Lewis/Loftus
//
//  Demonstrates the use of radio buttons.
//********************************************************************

import javax.swing.JFrame;

public class QuoteOptions
{
   //-----------------------------------------------------------------
   //  Creates and presents the program frame.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Quote Options");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      QuoteOptionsPanel panel = new QuoteOptionsPanel();
      frame.getContentPane().add(panel);

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson
Education, Inc.






```
//*****************                                    *************
//  QuoteOptions.
//
//  Demonstrates
//*****************                                    *************

import javax.swin

public class QuoteOptions
{
   //-------------                                     -------------
   //  Creates an
   //-------------                                     -------------
   public static
   {
      JFrame fram
      frame.setDe                                          SE);

      QuoteOption
      frame.getContentPane().add(panel);

      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  QuoteOptionsPanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of radio buttons.
//********************************************************************

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class QuoteOptionsPanel extends JPanel
{
   private JLabel quote;
   private JRadioButton comedy, philosophy, carpentry;
   private String comedyQuote, philosophyQuote, carpentryQuote;

   //-----------------------------------------------------------------
   //  Sets up a panel with a label and a set of radio buttons
   //  that control its text.
   //-----------------------------------------------------------------
   public QuoteOptionsPanel()
   {
      comedyQuote = "Take my wife, please.";
      philosophyQuote = "I think, therefore I am.";
      carpentryQuote = "Measure twice. Cut once.";

      quote = new JLabel(comedyQuote);
      quote.setFont(new Font("Helvetica", Font.BOLD, 24));
```

continue

continue

```
        comedy = new JRadioButton("Comedy", true);
        comedy.setBackground(Color.green);
        philosophy = new JRadioButton("Philosophy");
        philosophy.setBackground(Color.green);
        carpentry = new JRadioButton("Carpentry");
        carpentry.setBackground(Color.green);

        ButtonGroup group = new ButtonGroup();
        group.add(comedy);
        group.add(philosophy);
        group.add(carpentry);

        QuoteListener listener = new QuoteListener();
        comedy.addActionListener(listener);
        philosophy.addActionListener(listener);
        carpentry.addActionListener(listener);

        add(quote);
        add(comedy);
        add(philosophy);
        add(carpentry);

        setBackground(Color.green);
        setPreferredSize(new Dimension(300, 100));
    }
```

continue

continue

```
//*****************************************************************
//  Represents the listener for all radio buttons
//*****************************************************************
private class QuoteListener implements ActionListener
{
   //--------------------------------------------------------------
   //  Sets the text of the label depending on which radio
   //  button was pressed.
   //--------------------------------------------------------------
   public void actionPerformed(ActionEvent event)
   {
      Object source = event.getSource();

      if (source == comedy)
         quote.setText(comedyQuote);
      else
         if (source == philosophy)
            quote.setText(philosophyQuote);
         else
            quote.setText(carpentryQuote);
   }
}
}
```

Copyright © 2014 Pearson Education, Inc.

Drawing Techniques

- Conditionals and loops enhance our ability to generate interesting graphics

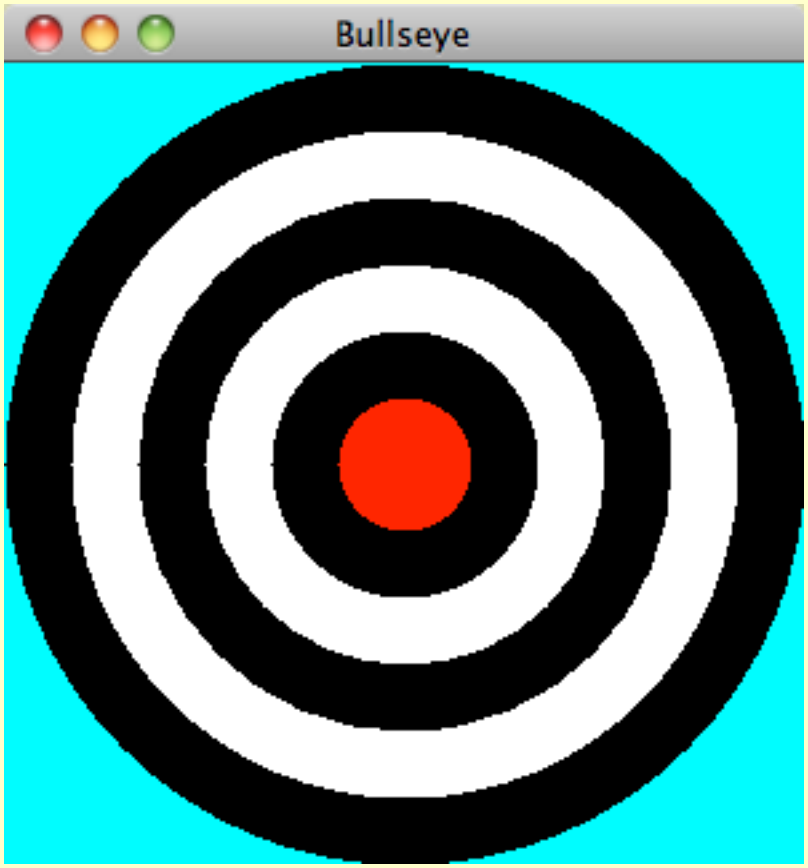
- See `Bullseye.java`
- See `BullseyePanel.java`

- See `Boxes.java`
- See `BoxesPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  Bullseye.java       Author: Lewis/Loftus
//
//  Demonstrates the use of loops to draw.
//********************************************************************

import javax.swing.JFrame;

public class Bullseye
{
   //-----------------------------------------------------------------
   //  Creates the main frame of the program.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Bullseye");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      BullseyePanel panel = new BullseyePanel();

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//*************                                          **************
//  Bullseye.j
//
//  Demonstrat
//*************                                          **************

import javax.s

public class B
{
   //---------                                          --------------
   //  Creates
   //---------                                          --------------
   public stat
   {
      JFrame f
      frame.se                                          OSE);

      Bullseye

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

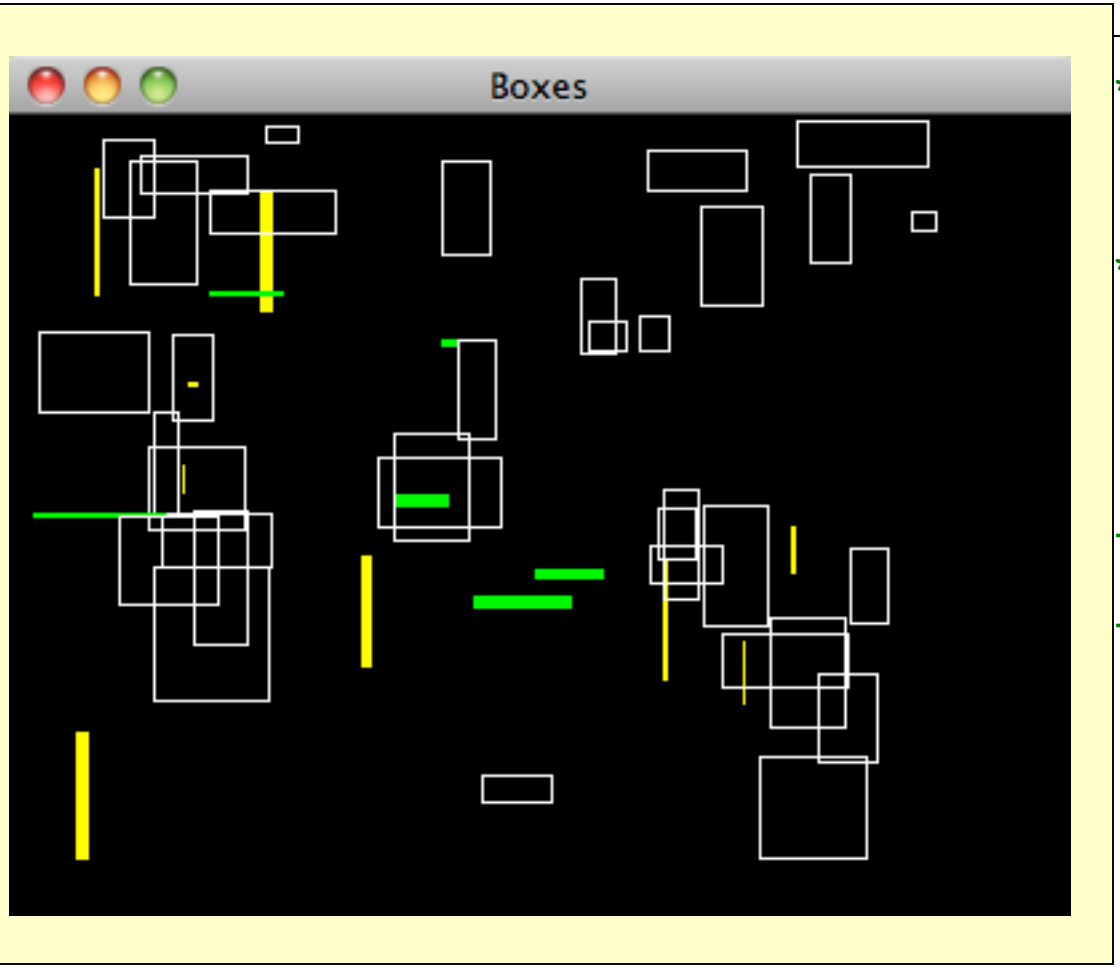
```
//********************************************************************
//  BullseyePanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of conditionals and loops to guide drawing.
//********************************************************************

import javax.swing.JPanel;
import java.awt.*;

public class BullseyePanel extends JPanel
{
   private final int MAX_WIDTH = 300, NUM_RINGS = 5, RING_WIDTH = 25;

   //-----------------------------------------------------------------
   //  Sets up the bullseye panel.
   //-----------------------------------------------------------------
   public BullseyePanel()
   {
      setBackground(Color.cyan);
      setPreferredSize(new Dimension(300,300));
   }
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
    //-----------------------------------------------------------------
    //  Paints a bullseye target.
    //-----------------------------------------------------------------
    public void paintComponent(Graphics page)
    {
        super.paintComponent (page);
        int x = 0, y = 0, diameter = MAX_WIDTH;
        page.setColor(Color.white);

        for (int count = 0; count < NUM_RINGS; count++)
        {
            if (page.getColor() == Color.black)  // alternate colors
                page.setColor(Color.white);
            else
                page.setColor Color.black);

            page.fillOval(x, y, diameter, diameter);

            diameter -= (2 * RING_WIDTH);
            x += RING_WIDTH;
            y += RING_WIDTH;
        }

        // Draw the red bullseye in the center
        page.setColor(Color.red);
        page.fillOval(x, y, diameter, diameter);
    }
}
```




```
//*********                                                    ********
//  Boxes.j
//
//  Demonst
//*********                                                    ********

import java

public clas
{
   //------                                                    --------
   //  Crea
   //------                                                    --------
   public s
   {
      JFrar
      frame

      Boxes

      frame.getContentPane().add(panel);
      frame.pack();
      frame.setVisible(true);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  BoxesPanel.java       Author: Lewis/Loftus
//
//  Demonstrates the use of conditionals and loops to guide drawing.
//********************************************************************

import javax.swing.JPanel;
import java.awt.*;
import java.util.Random;

public class BoxesPanel extends JPanel
{
   private final int NUM_BOXES = 50, THICKNESS = 5, MAX_SIDE = 50;
   private final int MAX_X = 350, MAX_Y = 250;
   private Random generator;

   //-----------------------------------------------------------------
   //  Sets up the drawing panel.
   //-----------------------------------------------------------------
   public BoxesPanel()
   {
      generator = new Random();

      setBackground(Color.black);
      setPreferredSize(new Dimension(400, 300));
   }
```

continue

Education, Inc.

continue

```
//-----------------------------------------------------------------
//  Paints boxes of random width and height in a random location.
//  Narrow or short boxes are highlighted with a fill color.
//-----------------------------------------------------------------
public void paintComponent(Graphics page)
{
   super.paintComponent(page);

   int x, y, width, height;

   for (int count = 0; count < NUM_BOXES; count++)
   {
      x = generator.nextInt(MAX_X) + 1;
      y = generator.nextInt(MAX_Y) + 1;

      width = generator.nextInt(MAX_SIDE) + 1;
      height = generator.nextInt(MAX_SIDE) + 1;
```

continue

Copyright © 2014 Pearson Education, Inc.

continue

```
            if (width <= THICKNESS)  // check for narrow box
            {
               page.setColor(Color.yellow);
               page.fillRect(x, y, width, height);
            }
            else
               if (height <= THICKNESS)  // check for short box
               {
                  page.setColor(Color.green);
                  page.fillRect(x, y, width, height);
               }
               else
               {
                  page.setColor(Color.white);
                  page.drawRect(x, y, width, height);
               }
         }
      }
   }
```

Copyright © 2014 Pearson Education, Inc.

Outline

- The `switch` Statement
- The Conditional Operator
- The `do` Statement
- The `for` Statement
- Drawing with Loops and Conditionals
- Dialog Boxes

Copyright © 2014 Pearson Education, Inc.

Dialog Boxes

- A *dialog box* is a window that appears on top of any currently active window
- It may be used to:
  - convey information
  - confirm an action
  - allow the user to enter data
  - pick a color
  - choose a file
- A dialog box usually has a specific, solitary purpose, and the user interaction with it is brief

Copyright © 2014 Pearson Education, Inc.

Dialog Boxes

- The `JOptionPane` class provides methods that simplify the creation of some types of dialog boxes
- See `EvenOdd.java`
- Specialized dialog boxes for choosing colors and files are covered in Chapter 9

Copyright © 2014 Pearson Education, Inc.

```
/********************************************************************
//  EvenOdd.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the JOptionPane class.
//********************************************************************

import javax.swing.JOptionPane;

public class EvenOdd
{
   //-----------------------------------------------------------------
   //  Determines if the value input by the user is even or odd.
   //  Uses multiple dialog boxes for user interaction.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      String numStr, result;
      int num, again;
```

continue

Copyright © 2014 Pearson Education, Inc.

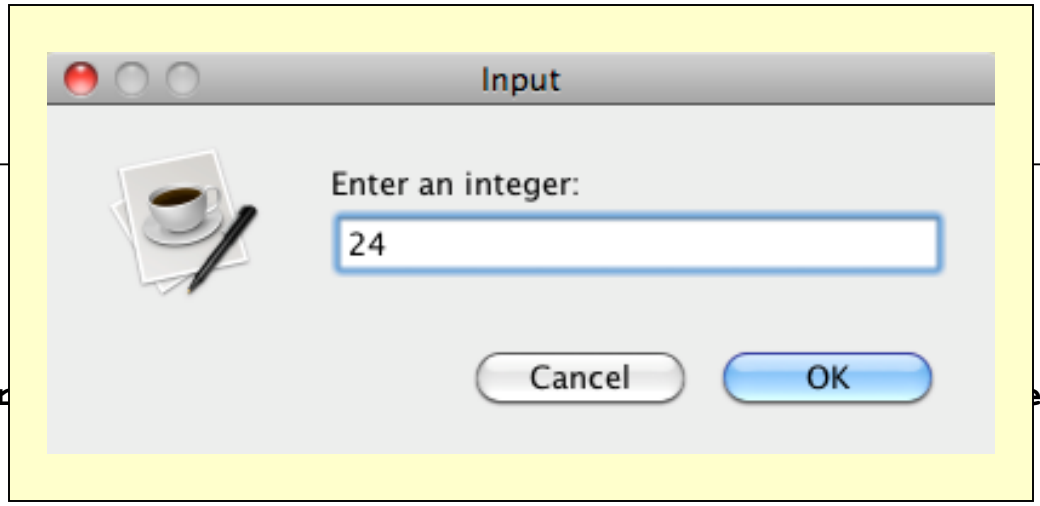
continue

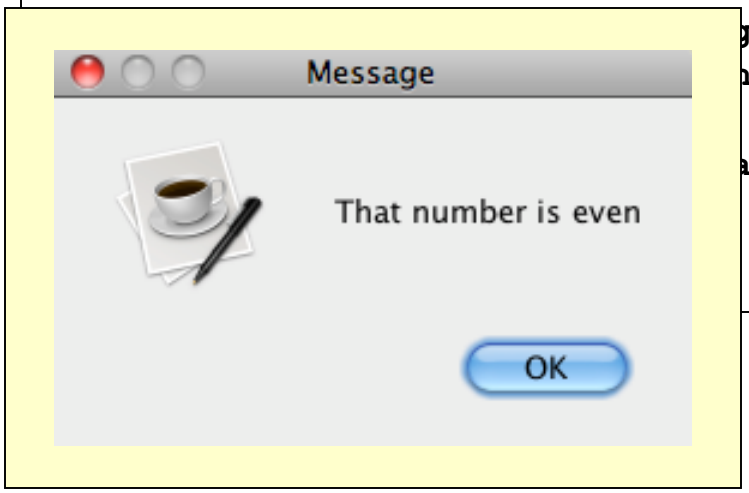
```
        do
        {
            numStr = JOptionPane.showInputDialog("Enter an integer: ");
            num = Integer.parseInt(numStr);

            result = "That number is " + ((num%2 == 0) ? "even" : "odd");

            JOptionPane.showMessageDialog(null, result);
            again = JOptionPane.showConfirmDialog(null, "Do Another?");
        }
        while (again == JOptionPane.YES_OPTION);
    }
}
```

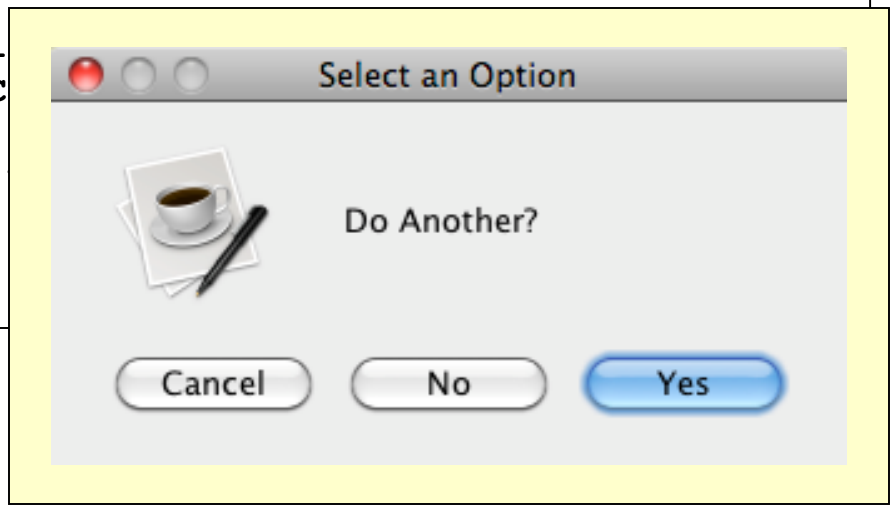
Copyright © 2014 Pearson Education, Inc.

continue

```
do
{
    numStr                                          eger: ");
    num =

    result = "That number is " + ((num%2 == 0) ? "even" : "odd");
```







Copyright © 2014 Pearson Education, Inc.

GUI Design

- We must remember that the goal of software is to help the user solve the problem
- To that end, the GUI designer should:
  - Know the user
  - Prevent user errors
  - Optimize user abilities
  - Be consistent
- Let's discuss each of these in more detail

Copyright © 2014 Pearson Education, Inc.

Know the User

- Knowing the user implies an understanding of:
  - the user's true needs
  - the user's common activities
  - the user's level of expertise in the problem domain and in computer processing
- We should also realize these issues may differ for different users
- Remember, to the user, the interface is the program

Copyright © 2014 Pearson Education, Inc.

Prevent User Errors

- Whenever possible, we should design user interfaces that minimize possible user mistakes
- We should choose the best GUI components for each task
- For example, in a situation where there are only a few valid options, using a menu or radio buttons would be better than an open text field
- Error messages should guide the user appropriately

Copyright © 2014 Pearson Education, Inc.

Optimize User Abilities

- Not all users are alike – some may be more familiar with the system than others
- Knowledgeable users are sometimes called *power users*
- We should provide multiple ways to accomplish a task whenever reasonable
  - "wizards" to walk a user through a process
  - short cuts for power users
- Help facilities should be available but not intrusive

Copyright © 2014 Pearson Education, Inc.

Be Consistent

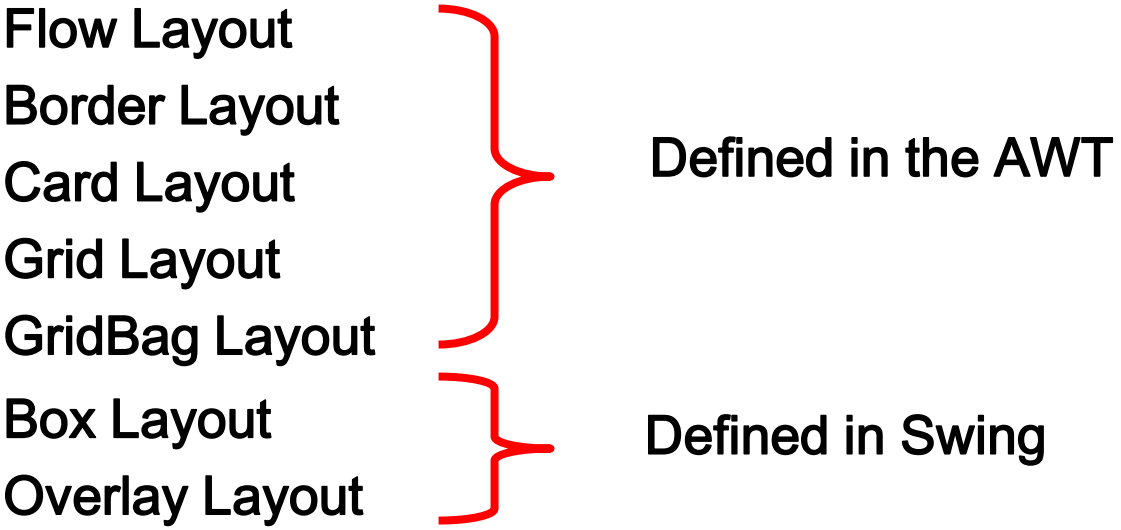
- Consistency is important – users get used to things appearing and working in certain ways
- Colors should be used consistently to indicate similar types of information or processing
- Screen layout should be consistent from one part of a system to another
- For example, error messages should appear in consistent locations

Copyright © 2014 Pearson Education, Inc.

Layout Managers

- A *layout manager* is an object that determines the way that components are arranged in a container

- There are several predefined layout managers defined in the Java API:



Copyright © 2014 Pearson Education, Inc.

Layout Managers

- Every container has a default layout manager, but we can explicitly set the layout manager as well
- Each layout manager has its own particular rules governing how the components will be arranged
- Some layout managers pay attention to a component's preferred size or alignment, while others do not
- A layout manager adjusts the layout as components are added and as containers are resized

Copyright © 2014 Pearson Education, Inc.

Layout Managers

- We can use the `setLayout` method of a container to change its layout manager:

```
JPanel panel = new JPanel();
panel.setLayout(new BorderLayout());
```

- The following example uses a *tabbed pane*, a container which permits one of several panes to be selected

- See `LayoutDemo.java`
- See `IntroPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  LayoutDemo.java       Authors: Lewis/Loftus
//
//  Demonstrates the use of flow, border, grid, and box layouts.
//********************************************************************

import javax.swing.*;

public class LayoutDemo
{
   //-----------------------------------------------------------------
   //  Sets up a frame containing a tabbed pane. The panel on each
   //  tab demonstrates a different layout manager.
   //-----------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Layout Manager Demo");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

continue

Copyright © 2014 Pearson
Education, Inc.

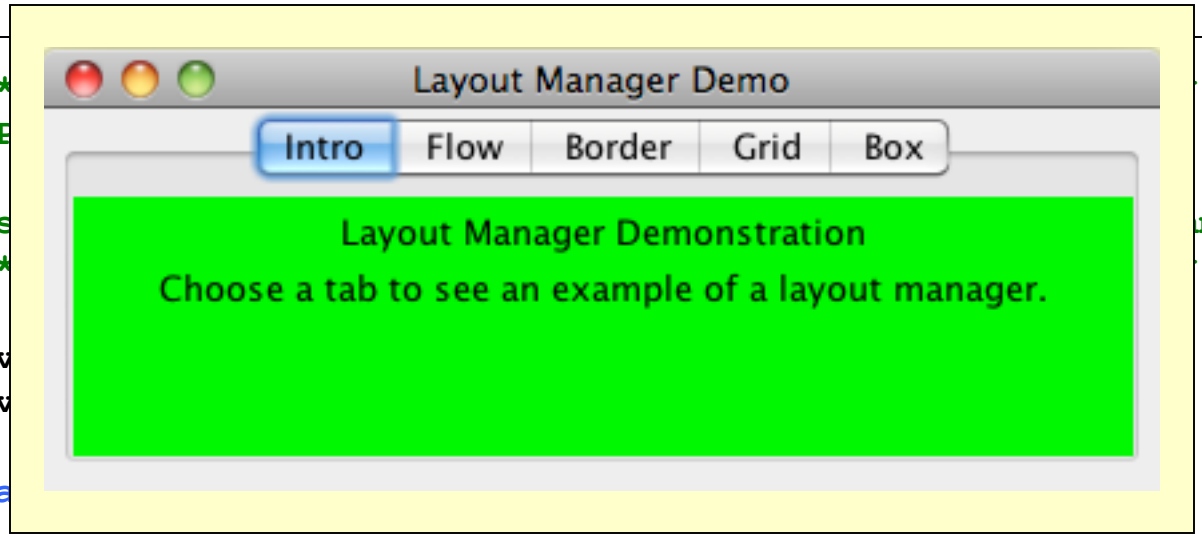
continue

```
        JTabbedPane tp = new JTabbedPane();
        tp.addTab("Intro", new IntroPanel());
        tp.addTab("Flow", new FlowPanel());
        tp.addTab("Border", new BorderPanel());
        tp.addTab("Grid", new GridPanel());
        tp.addTab("Box", new BoxPanel());

        frame.getContentPane().add(tp);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  IntroPanel.java       Authors: Lewis/Loftus
//
//  Represents the introduction panel for the LayoutDemo program.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class IntroPanel extends JPanel
{
   //-----------------------------------------------------------------
   //  Sets up this panel with two labels.
   //-----------------------------------------------------------------
   public IntroPanel()
   {
      setBackground(Color.green);

      JLabel l1 = new JLabel("Layout Manager Demonstration");
      JLabel l2 = new JLabel("Choose a tab to see an example of " +
                             "a layout manager.");

      add(l1);
      add(l2);
   }
}
```

Copyright © 2014 Pearson Education, Inc.




```
//********
//  IntroP                                                    *******
//
//  Repres                                                    m.
//********                                                   *******

import jav
import jav

public cla
{
   //-----------------------------------------------------------------
   //  Sets up this panel with two labels.
   //-----------------------------------------------------------------
   public IntroPanel()
   {
      setBackground(Color.green);

      JLabel l1 = new JLabel("Layout Manager Demonstration");
      JLabel l2 = new JLabel("Choose a tab to see an example of " +
                             "a layout manager.");

      add(l1);
      add(l2);
   }
}
```

Copyright © 2014 Pearson Education, Inc.

Flow Layout

- *Flow layout* puts as many components as possible on a row, then moves to the next row
- Components are displayed in the order they are added to the container
- Each row of components is centered horizontally by default, but could also be aligned left or right
- The horizontal and vertical gaps between the components can be explicitly set
- See `FlowPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  FlowPanel.java       Authors: Lewis/Loftus
//
//  Represents the panel in the LayoutDemo program that demonstrates
//  the flow layout manager.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class FlowPanel extends JPanel
{
   //-----------------------------------------------------------------
   //  Sets up this panel with some buttons to show how flow layout
   //  affects their position.
   //-----------------------------------------------------------------
   public FlowPanel()
   {
      setLayout(new FlowLayout());

      setBackground(Color.green);
```

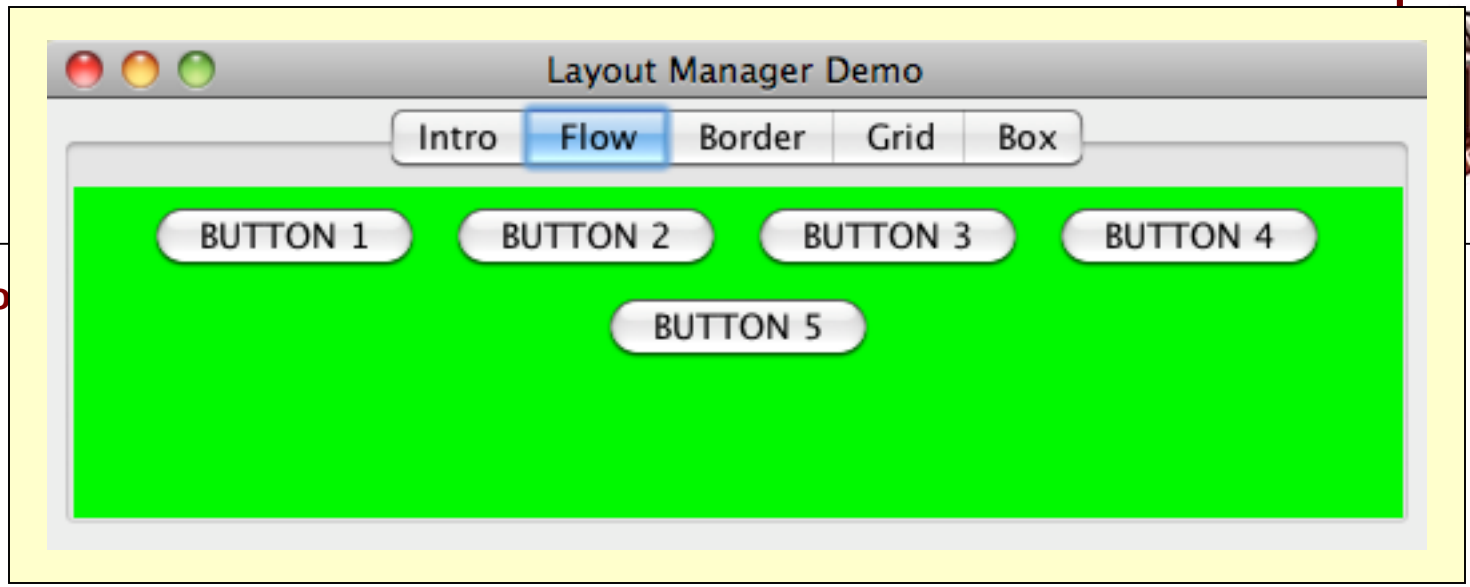
continue

Copyright © 2014 Pearson Education, Inc.

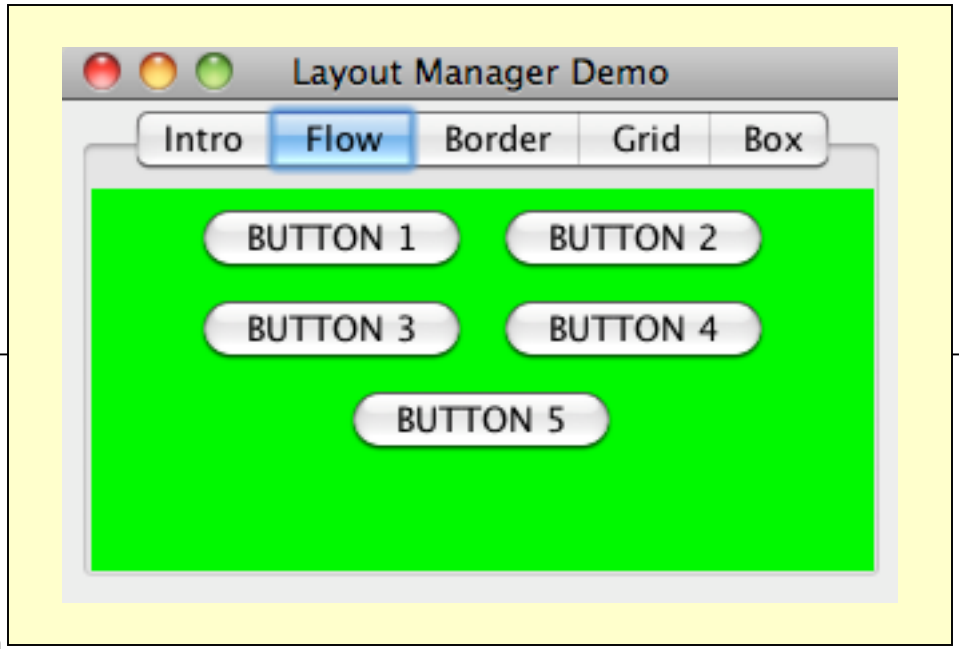
continue

```
        JButton b1 = new JButton("BUTTON 1");
        JButton b2 = new JButton("BUTTON 2");
        JButton b3 = new JButton("BUTTON 3");
        JButton b4 = new JButton("BUTTON 4");
        JButton b5 = new JButton("BUTTON 5");

        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
    }
}
```

Copyright © 2014 Pearson Education, Inc.



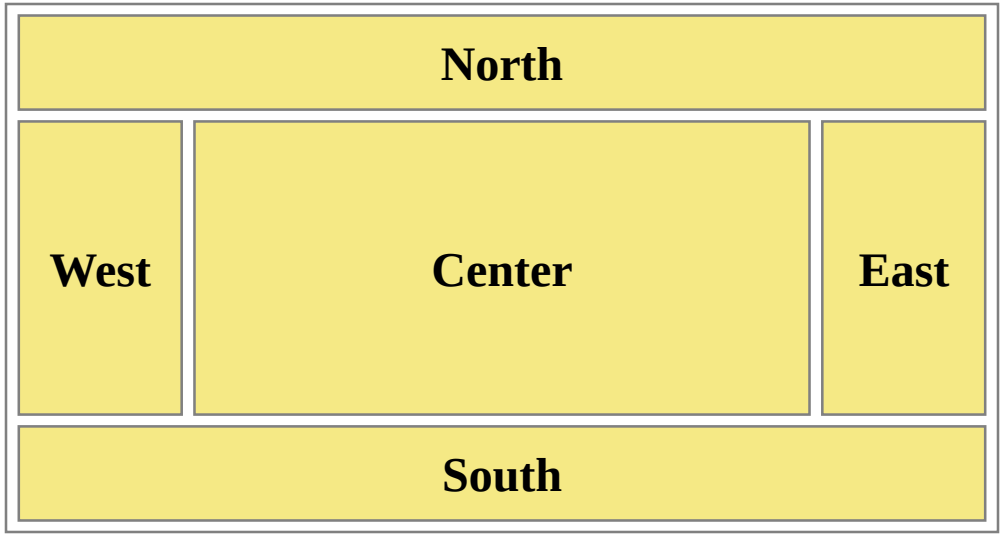

```
co
        add(b1)
        add(b2)
        add(b3)
        add(b4)
        add(b5)
    }
}
```



Copyright © 2014 Pearson Education, Inc.

Border Layout

- A *border layout* defines five areas into which components can be added



Copyright © 2014 Pearson Education, Inc.

Border Layout

- Each area displays one component (which could be a container such as a `JPanel`)
- Each of the four outer areas enlarges as needed to accommodate the component added to it
- If nothing is added to the outer areas, they take up no space and other areas expand to fill the void
- The center area expands to fill space as needed
- See `BorderPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  BorderPanel.java       Authors: Lewis/Loftus
//
//  Represents the panel in the LayoutDemo program that demonstrates
//  the border layout manager.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class BorderPanel extends JPanel
{
   //-----------------------------------------------------------------
   //  Sets up this panel with a button in each area of a border
   //  layout to show how it affects their position, shape, and size.
   //-----------------------------------------------------------------
   public BorderPanel()
   {
      setLayout(new BorderLayout());

      setBackground(Color.green);
```

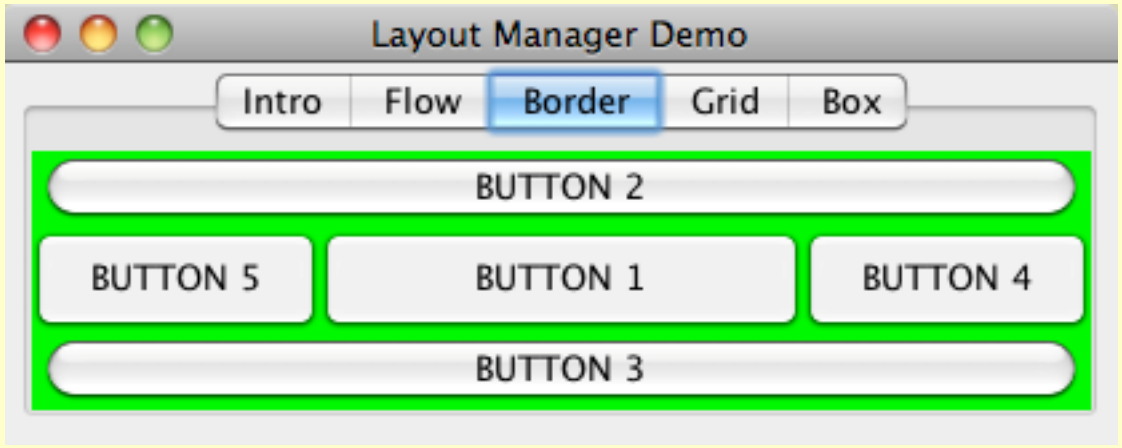
continue

Copyright © 2014 Pearson Education, Inc.

continue

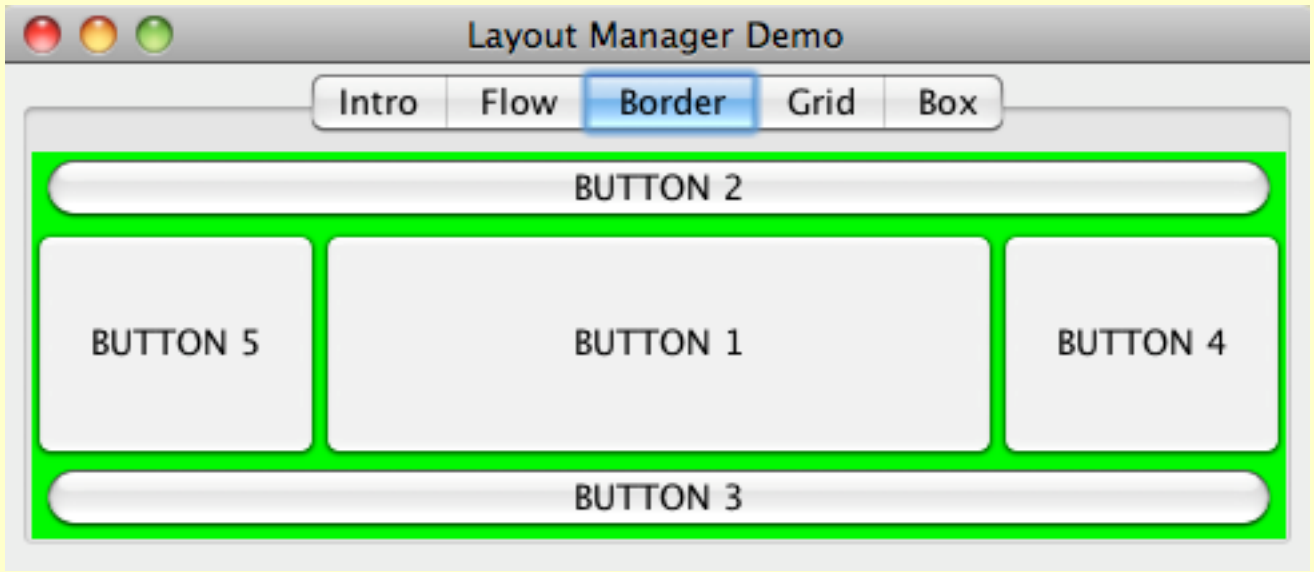
```
        JButton b1 = new JButton("BUTTON 1");
        JButton b2 = new JButton("BUTTON 2");
        JButton b3 = new JButton("BUTTON 3");
        JButton b4 = new JButton("BUTTON 4");
        JButton b5 = new JButton("BUTTON 5");

        add(b1, BorderLayout.CENTER);
        add(b2, BorderLayout.NORTH);
        add(b3, BorderLayout.SOUTH);
        add(b4, BorderLayout.EAST);
        add(b5, BorderLayout.WEST);
    }
}
```

Copyright © 2014 Pearson Education, Inc.



continue

```
        JBu
        JBu
        JBu
        JButton b4 = new JButton("BUTTON 4");
        JButton b5 = new JButton("BUTTON 5");
```




```
    }
}
```

Copyright © 2014 Pearson Education, Inc.

Grid Layout

- A *grid layout* presents a container's components in a rectangular grid of rows and columns
- One component is placed in each cell of the grid, and all cells have the same size
- Components fill the grid from left-to-right and top-to-bottom (by default)
- The size of each cell is determined by the overall size of the container
- See `GridPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  GridPanel.java       Authors: Lewis/Loftus
//
//  Represents the panel in the LayoutDemo program that demonstrates
//  the grid layout manager.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class GridPanel extends JPanel
{
   //-----------------------------------------------------------------
   //  Sets up this panel with some buttons to show how grid
   //  layout affects their position, shape, and size.
   //-----------------------------------------------------------------
   public GridPanel()
   {
      setLayout(new GridLayout(2, 3));

      setBackground(Color.green);
```

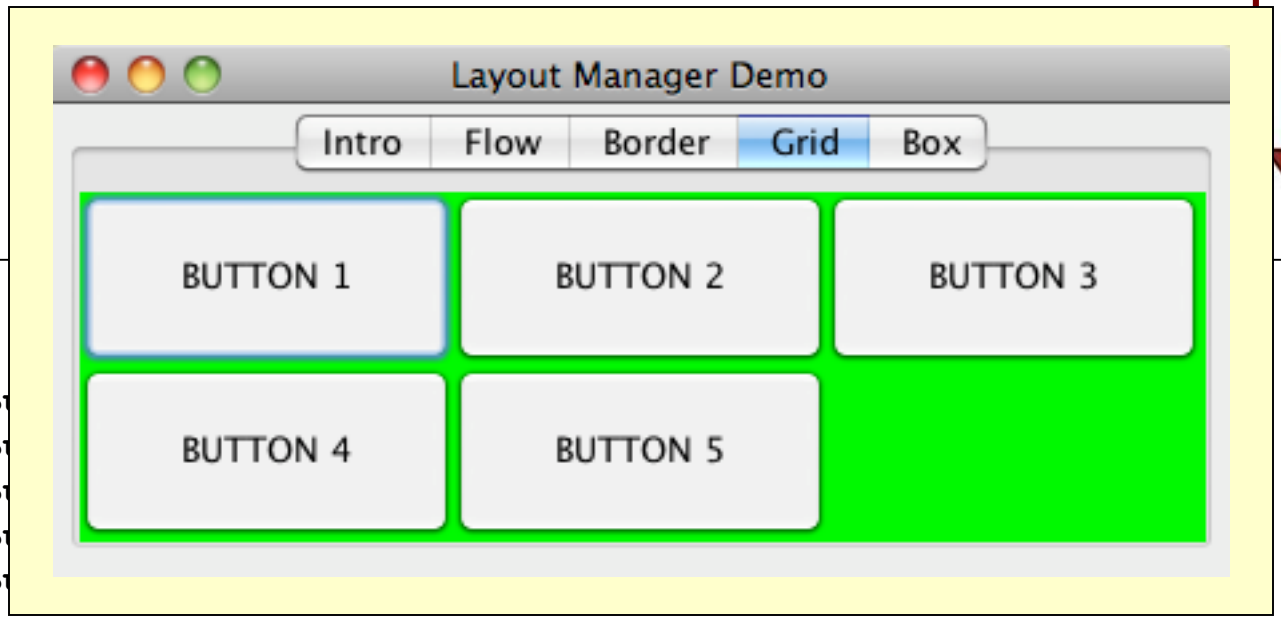
continue

Copyright © 2014 Pearson Education, Inc.

continue

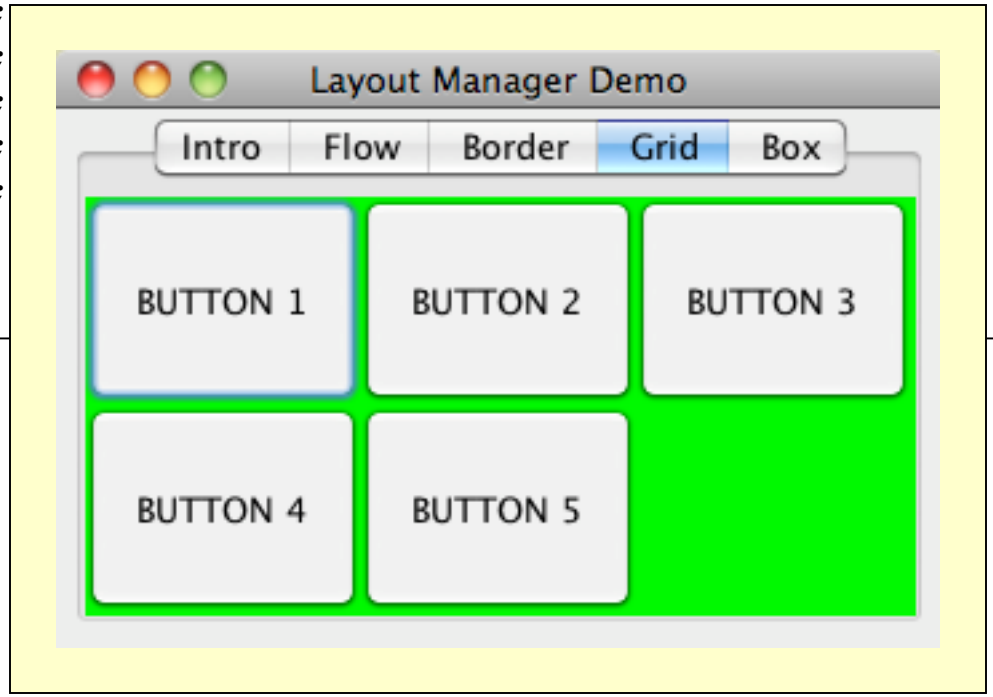
```
        JButton b1 = new JButton("BUTTON 1");
        JButton b2 = new JButton("BUTTON 2");
        JButton b3 = new JButton("BUTTON 3");
        JButton b4 = new JButton("BUTTON 4");
        JButton b5 = new JButton("BUTTON 5");

        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
    }
}
```

Copyright © 2014 Pearson Education, Inc.



continue

```
        JBu
        JBu
        JBu
        JBu
        JBu

        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
    }
}
```



Copyright © 2014 Pearson Education, Inc.

Box Layout

- A *box layout* organizes components horizontally (in one row) or vertically (in one column)
- Components are placed top-to-bottom or left-to-right in the order in which they are added to the container
- By combining multiple containers using box layout, many different configurations can be created
- Multiple containers with box layouts are often preferred to one container that uses the more complicated gridbag layout manager

Copyright © 2014 Pearson Education, Inc.

Box Layout

- *Invisible components* can be added to a box layout container to take up space between components
  - *Rigid areas* have a fixed size
  - *Glue* specifies where excess space should go
- Invisible components are created using these methods of the `Box` class:

```
createRigidArea(Dimension d)
createHorizontalGlue()
createVerticalGlue()
```

- See `BoxPanel.java`

Copyright © 2014 Pearson Education, Inc.

```
//********************************************************************
//  BoxPanel.java       Authors: Lewis/Loftus
//
//  Represents the panel in the LayoutDemo program that demonstrates
//  the box layout manager.
//********************************************************************

import java.awt.*;
import javax.swing.*;

public class BoxPanel extends JPanel
{
   //-----------------------------------------------------------------
   //  Sets up this panel with some buttons to show how a vertical
   //  box layout (and invisible components) affects their position.
   //-----------------------------------------------------------------
   public BoxPanel()
   {
      setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));

      setBackground(Color.green);
```

continue

Copyright © 2014 Pearson Education, Inc.

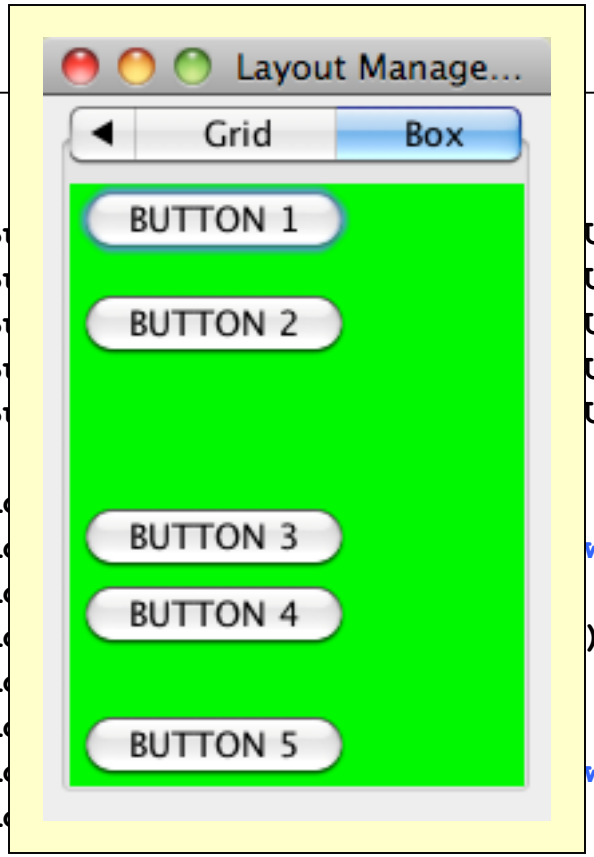
continue

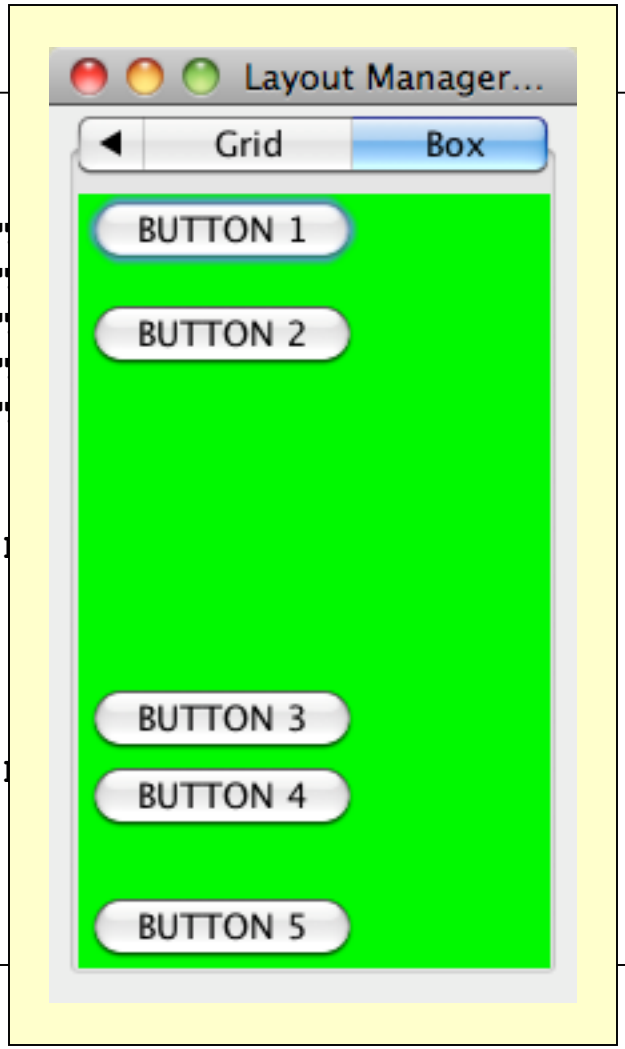
```
        JButton b1 = new JButton("BUTTON 1");
        JButton b2 = new JButton("BUTTON 2");
        JButton b3 = new JButton("BUTTON 3");
        JButton b4 = new JButton("BUTTON 4");
        JButton b5 = new JButton("BUTTON 5");

        add(b1);
        add(Box.createRigidArea(new Dimension(0, 10)));
        add(b2);
        add(Box.createVerticalGlue());
        add(b3);
        add(b4);
        add(Box.createRigidArea(new Dimension(0, 20)));
        add(b5);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

continue

```
            JBu                                                    UT"
            JBu                                                    UT"
            JBu                                                    UT"
            JBu                                                    UT"
            JBu                                                    UT"

            ado
            ado                                                    w
            ado
            ado                                                    ) )
            ado
            ado
            ado                                                    w
            ado
        }
}
```





Copyright © 2014 Pearson Education, Inc.

Borders

- A *border* can be put around any Swing component to define how the edges of the component should be drawn
- Borders can be used effectively to group components visually
- The `BorderFactory` class contains several static methods for creating border objects
- A border is applied to a component using the `setBorder` method

Copyright © 2014 Pearson Education, Inc.

Borders

- An *empty border*
  - buffers the space around the edge of a component
  - otherwise has no visual effect
- A *line border*
  - surrounds the component with a simple line
  - the line's color and thickness can be specified
- An *etched border*
  - creates the effect of an etched groove around a component
  - uses colors for the highlight and shadow

Copyright © 2014 Pearson Education, Inc.

Borders

- A *bevel border*
  - can be raised or lowered
  - uses colors for the outer and inner highlights and shadows
- A *titled border*
  - places a title on or around the border
  - the title can be oriented in many ways
- A *matte border*
  - specifies the sizes of the top, left, bottom, and right edges of the border separately
  - uses either a solid color or an image

Copyright © 2014 Pearson Education, Inc.

Borders

- A *compound border*
  - is a combination of two borders
  - one or both of the borders can be a compound border
- See `BorderDemo.java`

Copyright © 2014 Pearson Education, Inc.

```
//*****************************************************************
//  BorderDemo.java       Authors: Lewis/Loftus
//
//  Demonstrates the use of various types of borders.
//*****************************************************************

import java.awt.*;
import javax.swing.*;
import javax.swing.border.*;

public class BorderDemo
{
   //--------------------------------------------------------------
   //  Creates several bordered panels and displays them.
   //--------------------------------------------------------------
   public static void main(String[] args)
   {
      JFrame frame = new JFrame("Border Demo");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JPanel panel = new JPanel();
      panel.setLayout(new GridLayout(0, 2, 5, 10));
      panel.setBorder(BorderFactory.createEmptyBorder(8, 8, 8, 8));

      JPanel p1 = new JPanel();
      p1.setBorder(BorderFactory.createLineBorder(Color.red, 3));
      p1.add(new JLabel("Line Border"));
      panel.add(p1);
```

continue

continue

```
JPanel p2 = new JPanel();
p2.setBorder(BorderFactory.createEtchedBorder());
p2.add(new JLabel("Etched Border"));
panel.add(p2);

JPanel p3 = new JPanel();
p3.setBorder(BorderFactory.createRaisedBevelBorder());
p3.add(new JLabel("Raised Bevel Border"));
panel.add(p3);

JPanel p4 = new JPanel();
p4.setBorder(BorderFactory.createLoweredBevelBorder());
p4.add(new JLabel("Lowered Bevel Border"));
panel.add(p4);

JPanel p5 = new JPanel();
p5.setBorder(BorderFactory.createTitledBorder("Title"));
p5.add(new JLabel("Titled Border"));
panel.add(p5);

JPanel p6 = new JPanel();
TitledBorder tb = BorderFactory.createTitledBorder("Title");
tb.setTitleJustification(TitledBorder.RIGHT);
p6.setBorder(tb);
p6.add(new JLabel("Titled Border(right)"));
panel.add(p6);
```

continue

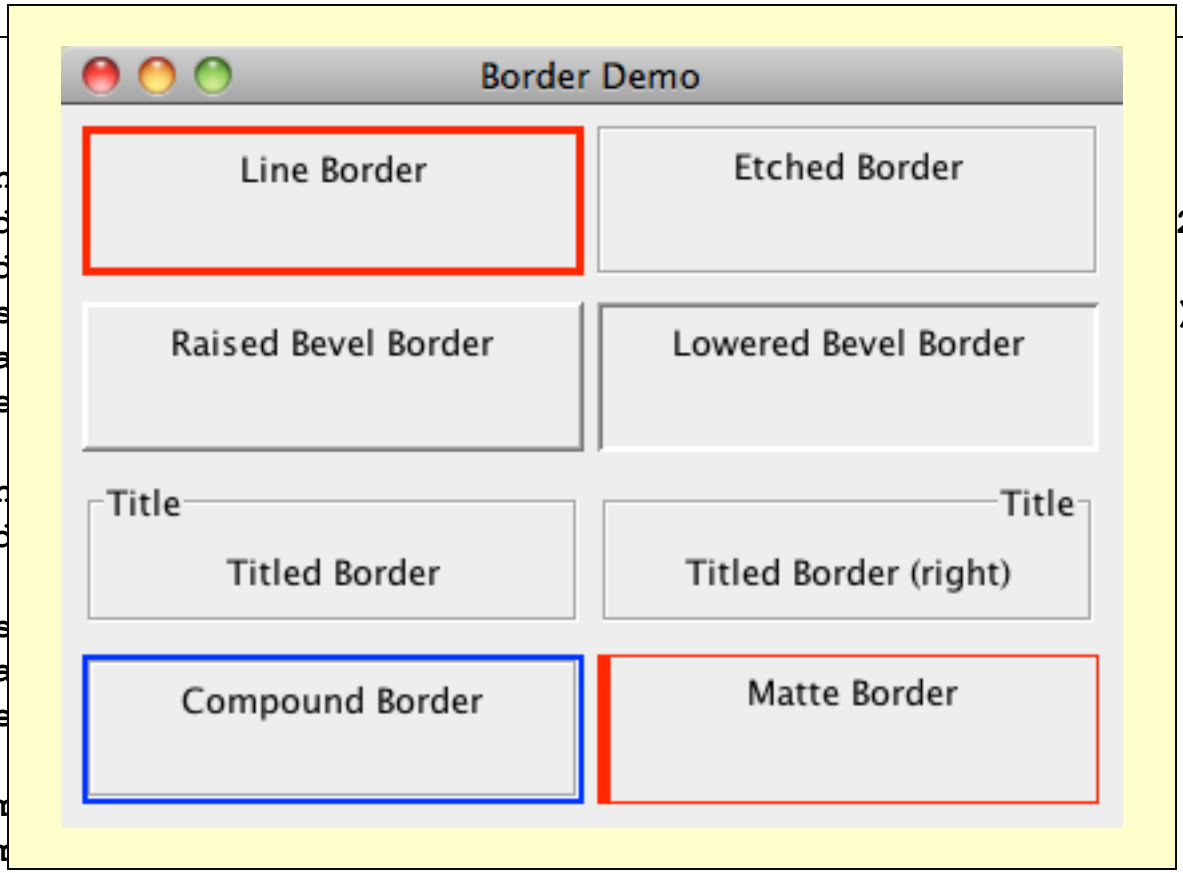
continue

```
        JPanel p7 = new JPanel();
        Border b1 = BorderFactory.createLineBorder(Color.blue, 2);
        Border b2 = BorderFactory.createEtchedBorder();
        p7.setBorder (BorderFactory.createCompoundBorder(b1, b2));
        p7.add (new JLabel("Compound Border"));
        panel.add(p7);

        JPanel p8 = new JPanel();
        Border mb = BorderFactory.createMatteBorder(1, 5, 1, 1,
                                                    Color.red);
        p8.setBorder(mb);
        p8.add(new JLabel("Matte Border"));
        panel.add(p8);

        frame.getContentPane().add(panel);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Copyright © 2014 Pearson Education, Inc.

continue

```
        JPar
        Bord                                                    2);
        Bord
        p7.s                                                    ));
        p7.a
        pane

        JPar
        Bord

        p8.s
        p8.a
        pane

        fram
        fram
        frame.setVisible(true);
    }
}
```



Copyright © 2014 Pearson Education, Inc.